Pipeline Proposal

1. Introduction
2. Studio Resources
3. Setup And Handle the Studio Resources
4. DCCs to use in different steps
5. Work Flow
6. Detailed Workflow with specific step
7. Folder Structure
8. Versioning
9. Published Steps Data Types
10. Core Pipeline Production Tools
11. Pipeline Resources
12. Pipeline Packages (Repository)
13. Mailing and Security
14. Example of Automation that connect with two dependency steps

1. Introduction

I believe you are looking for a strong and stable Pipeline. That should be dynamically working on the different shows.

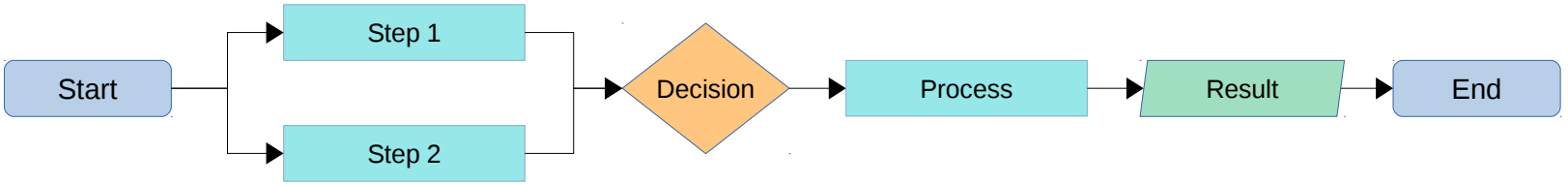
For consistency of the pipeline depends on the workflow of the studio, if the studio is following a good or bad workflow not a matter, but the first studio needs to finalize the workflow. If the studio changes the workflow in between specific projects that make more problems in the pipeline (data management). This is the main reason if any problem pop-up in the pipeline.

We can think about two kinds of workflow

- **Linear**
- **Non-Linear**

I recommend Linear workflow for Animation production and Non-Linear for VFX production.

Here, we can look at the Linear workflow. What I mean by linear? Data travel with a specific hierarchy structure. I think the example makes sense.



Development of production pipeline is not a small undertaking and success, it depends on many things. However, the most important part of it to establish well-defined guidelines, practices, and disciplines. Some of the disciplines mentioned below are

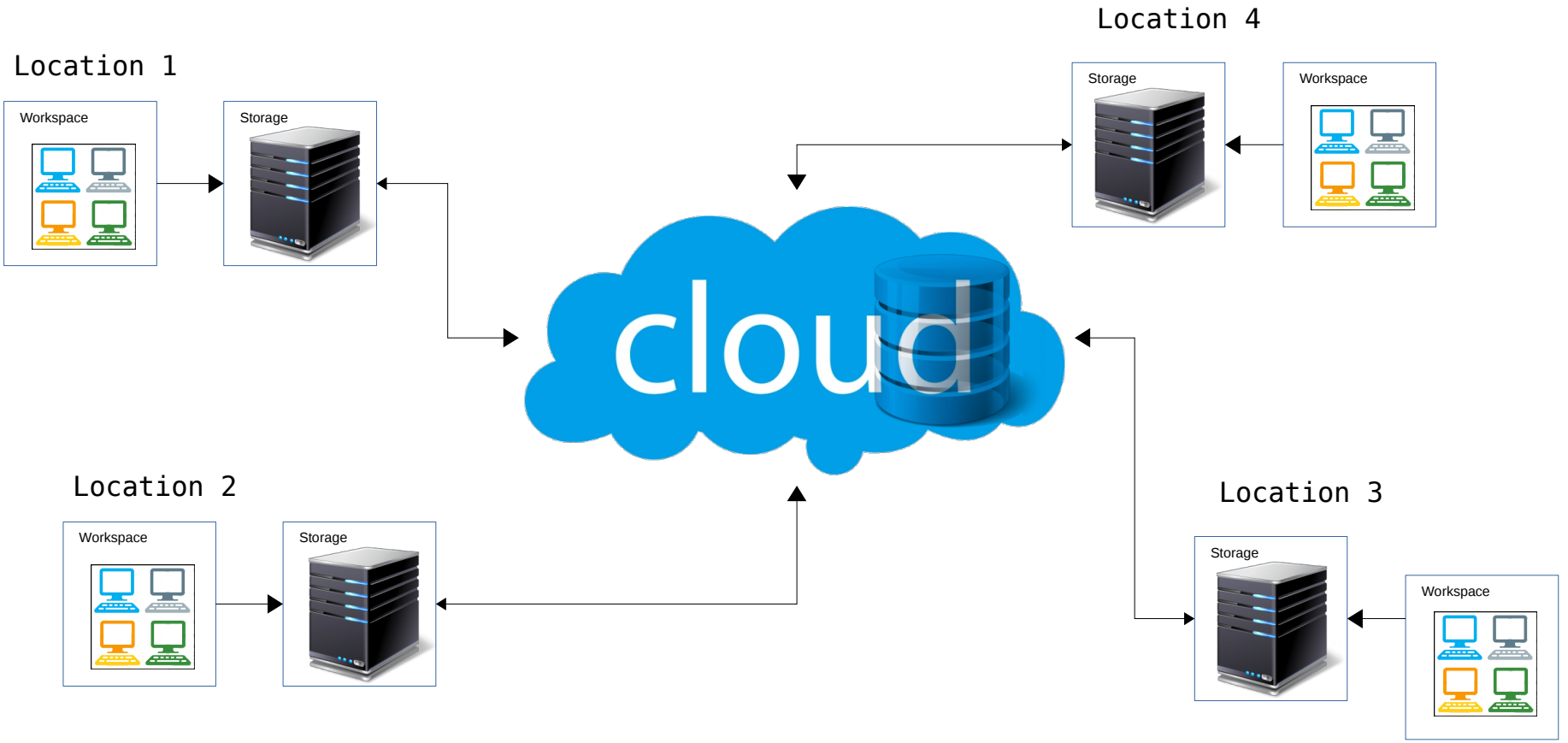
- **Consistency**
- **Strong Decision**
- **Documentation**
- **Communication**
- **Planing**

2. Studio Resources

Here, our resources are local storage with different locations and one common cloud storage. Based on this resource I am going to propose the linear workflow and pipeline for animation.

In our local and cloud, we need to maintain the same data structure. This is important think because this will help our all location works are going smooth. Our Pipeline tools will help to maintain this data structure on all local and cloud. And this should be an automation process.

We can plug multiple locations into the cloud.



Server, I recommend configuring the one Global server. But the IT department needs to take care of server configure. I can help you as per our requirements.

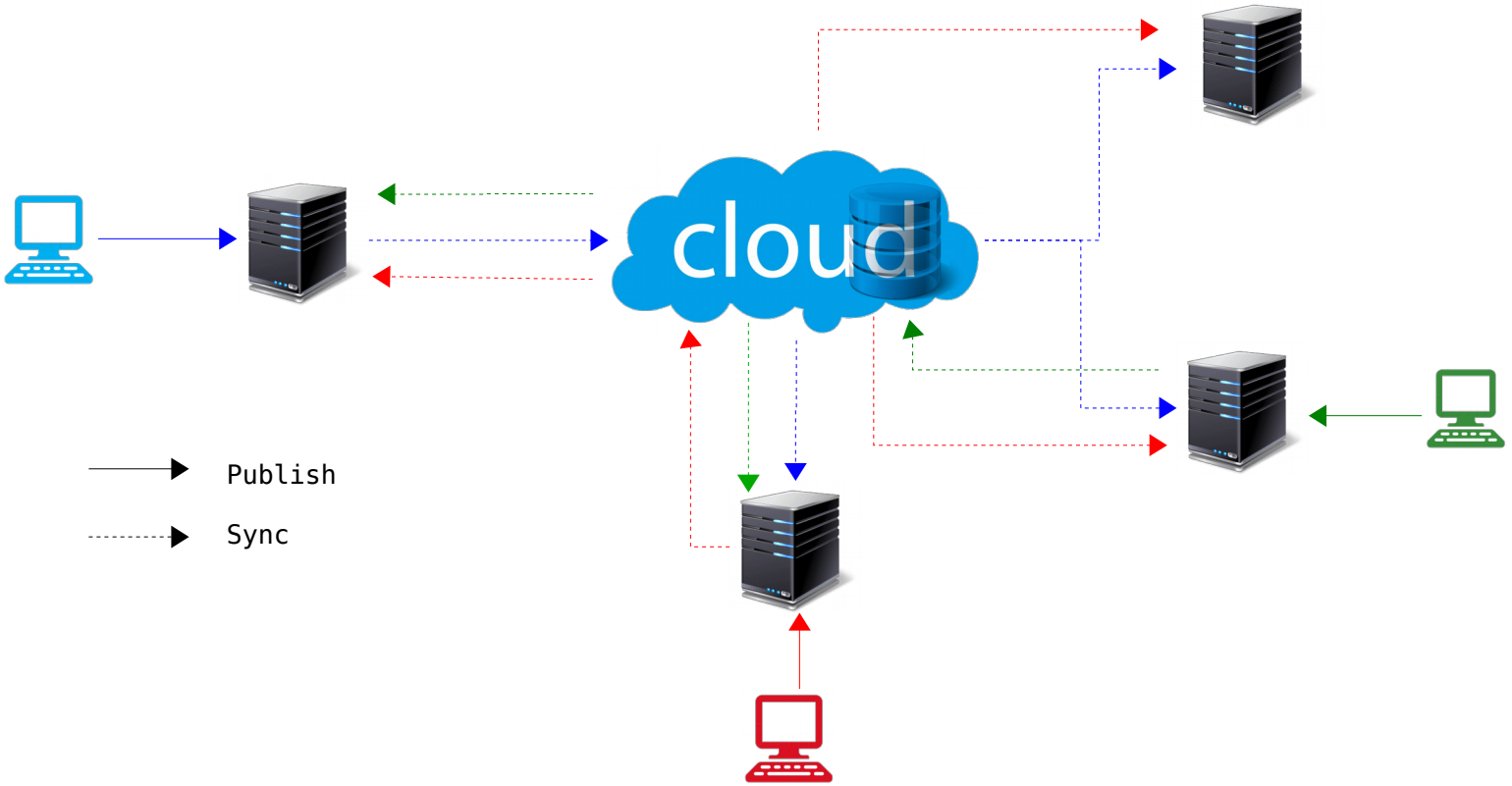
3. Setup And Handle the Studio Resources

Workspace is the local user pc. For each location workspace, I recommend following the same discipline, to avoid complexity. The pipeline team can handle to make the discipline for each location. For example ENV config, software config, etc.

Storage is the local location storage unit, workspace, and storage are dependent on each other. All configs, show data, and pipeline setups are set up on the local location storage. And I recommend following the same discipline in each location storage. And to avoid complexity. The pipeline team can handle to make the discipline for each location storage.

Cloud is the Global storage unit, this is our final valid data storage unit. And this Global storage unit is in sync with each location storage. For example, Once local workspace publishes happened, first it will store local storage and that new data will sync with cloud storage. Ultimately local and Cloud storage data identically the same. If this process happened that will sync with other local storage. The result is identically the same in all local storage and cloud. At the same time pipeline can control, all the sync processes. Before syncing, the pipeline can valid the data as well. Because validation is very important, to avoid garbage data.

Look at the below flowchart to get more details,



How to sync from local to cloud and cloud to local?

Here from local to cloud, name it as push-sync and from cloud to local pull-sync

Push-Sync: In this sync, we can add to build-in publish tool and it should be optional and we can provide a global sync tool for push sync, this tool can push a group of published data.

Pull-Sync: here we can use the same Global sync tool to pull the latest cloud published data to local. The production team needs to take care Global Sync tool. Make sure all the data are sync well. The tool will help with what needs to be sync, it's not a hard task.

Either we can do auto sync, I recommend initially we can go with a tool, and later we can think about auto-sync.

Here I did not mention how to manage **freelance works**. So the freelances, we can pass some packing tools to pack their data for submission. And once we receive the packed data we can unpack the data with the help of the packing tool. After unpacking the data distribution happened based on as per our discipline. If the unpacked data is approved, the team lead or respective people can publish. Once it publish that data under our control.

Core Tools.

- **DCC Launcher**
  This is the show config tool and launch the respective software with proper ENV. This tool can handle utilize the pipeline features. If it is a different location, the pipeline can track and control the user tasks.

- **Publish Tool**
  To publish the valid data to local and cloud.

- **Global Sync Tool**
  Publish data sync to local and local to cloud

- **Packing and unpack Tool**
  To handle the Freelancer data

4. DCCs to use in different steps

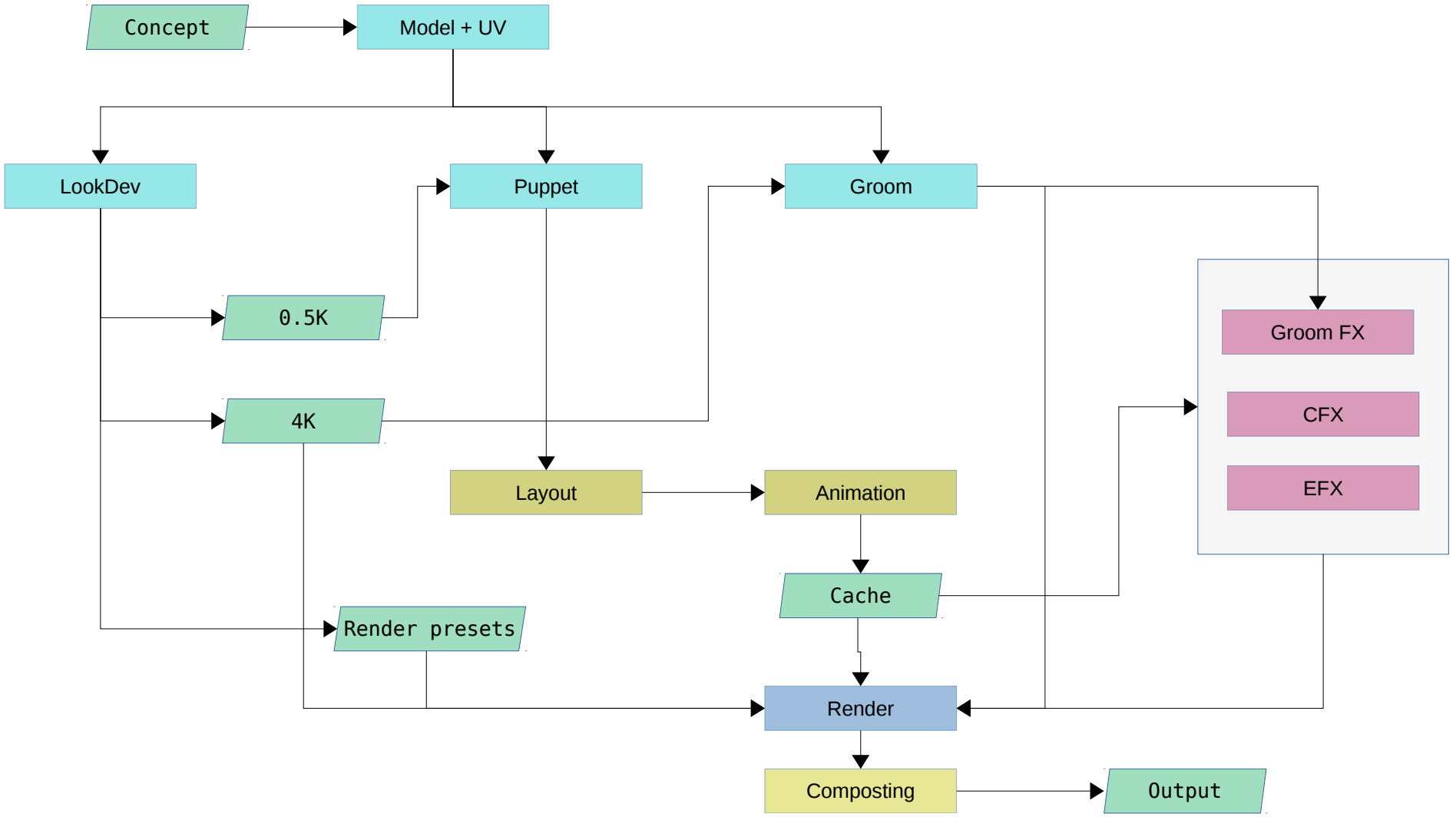
I recommend mixing open-source software in production, the reason is we can save the investment in the software license. But it's up to your team's legacy. Please look at the below column some combinations are mentioned. This we can discuss with your team, what should be convenient for them.

Steps	1	2	3
Editorial	Adobe Apps	Blender	Adobe Apps or Blender
Modeling	Maya	Blender	Blender
Lookdev	Mari, Adobe, Katana	Blender, Gimp, Gaffer	Mari, Adobe, Katana
Rigging	Maya	Blender	Maya
Animation	Maya	Blender	Maya
Rendering	Katana Deadline	Gaffer	Maya
Composting	Nuke	Natron or Blender	Nuke

Common	Tools
Operating System	Linux CentOS-7 64Bit
Pipeline	Python 2.7 or 3, PySide, Git, PyCharm or Eclipse
Bug Tracking Tool	Jira/Bugzilla/Redmine
Project Management Tool	Shotgun
Documentation Tool	Wiki

5. Work flow

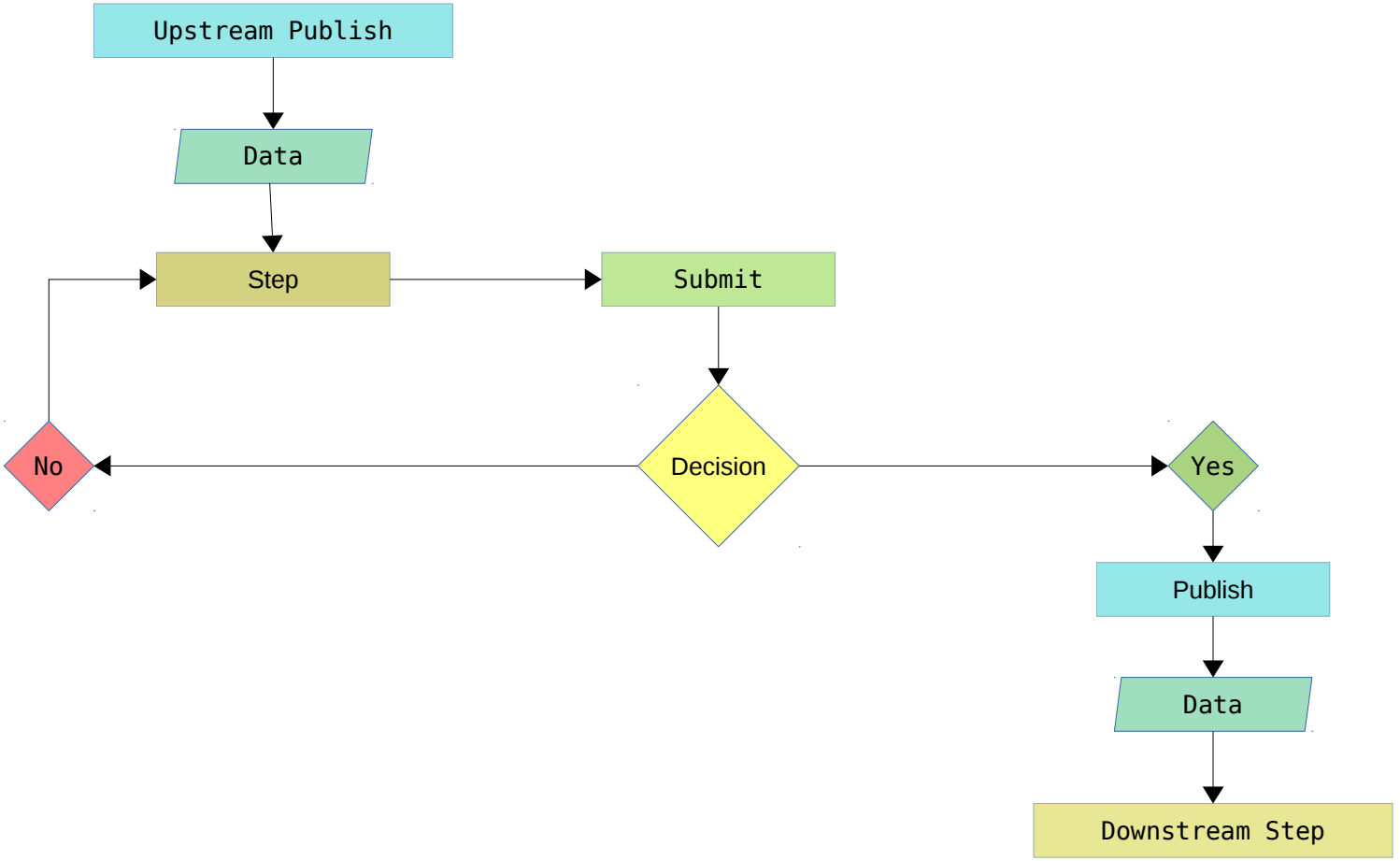
Show (project) Workflow: this is the basic setup, but we can update based on your team requirements, legacy and budget. I am looking at your team's suggestions for designing the workflow. Based on the workflow only I can design and develop the core pipeline packages and modules. Just go with the below workflow. Remember simple is better than complex.



6. Detailed Work flow with specific step

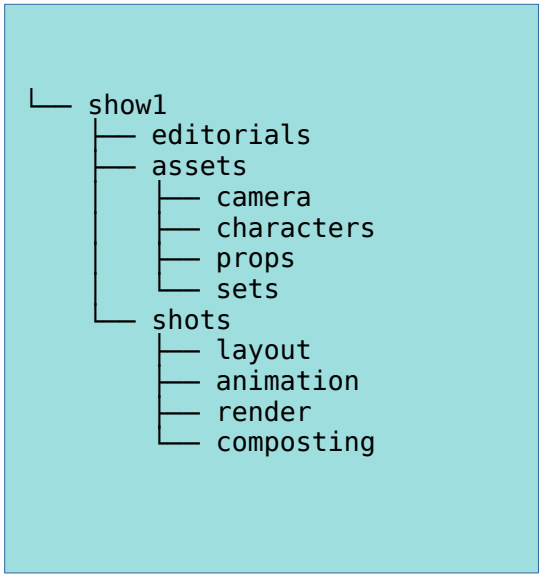
Here look at the below flow chart, data is coming from upstream step publish, and once it comes to the current step user need to submit his/her file, If it is not approved its go back to the user with messages, and he/she fix the bug and need to submit again, then is it approved, it goes to publish process. And publish data has to block (no modification). If you want to add on the existing publish, he/she can publish as a next version.

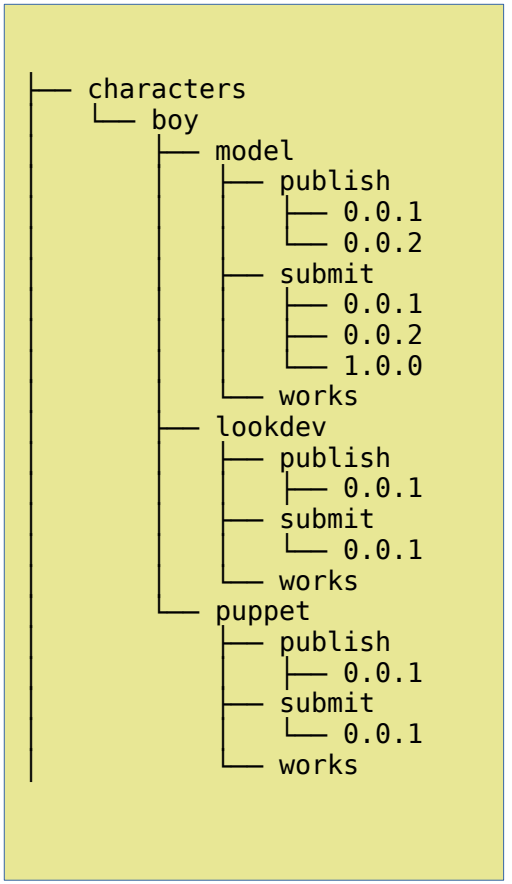
The advantage of submitting, we can avoid multiple publish processes. Submitted data are temporary files, once it's approved we can delete it. Either each correction we need to make each publish. I recommend this publish data should be valid and should pass to downstream steps. For the review proposes, we can use submit process instead of publishing. This one of the best practices to optimize our storage.

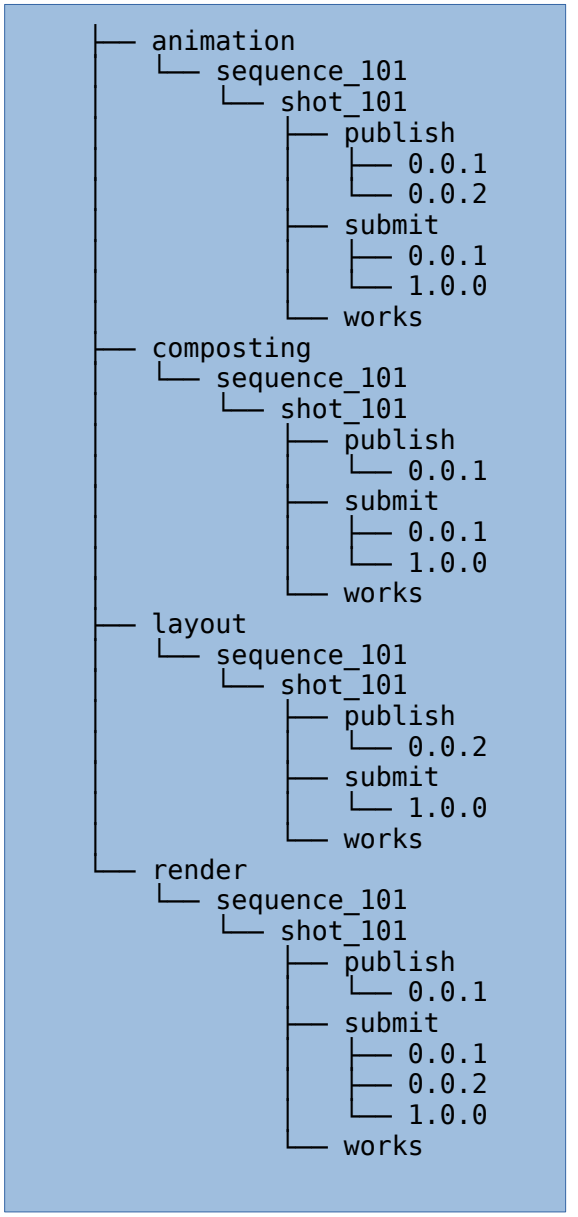


7. Folder Structure

Is it never matter what kind of folder structure we are following, because all valid data we handle through the pipeline tools, but pretty and standard folder structures avoid much confusion and easy to find the data. I propose the below folder structure. This is the simple, efficient, and standard folder structure. The editorials I am skipping because I would like to discuss more with supervisor.







8. Versioning

For versions, I prefer Semantic Versioning.

- MAJOR - when you make incompatible changes.
- MINOR - when you add functionality in a backward-compatible manner
- PATCH - when you make backward-compatible bug fixes.

More details https://semver.org/

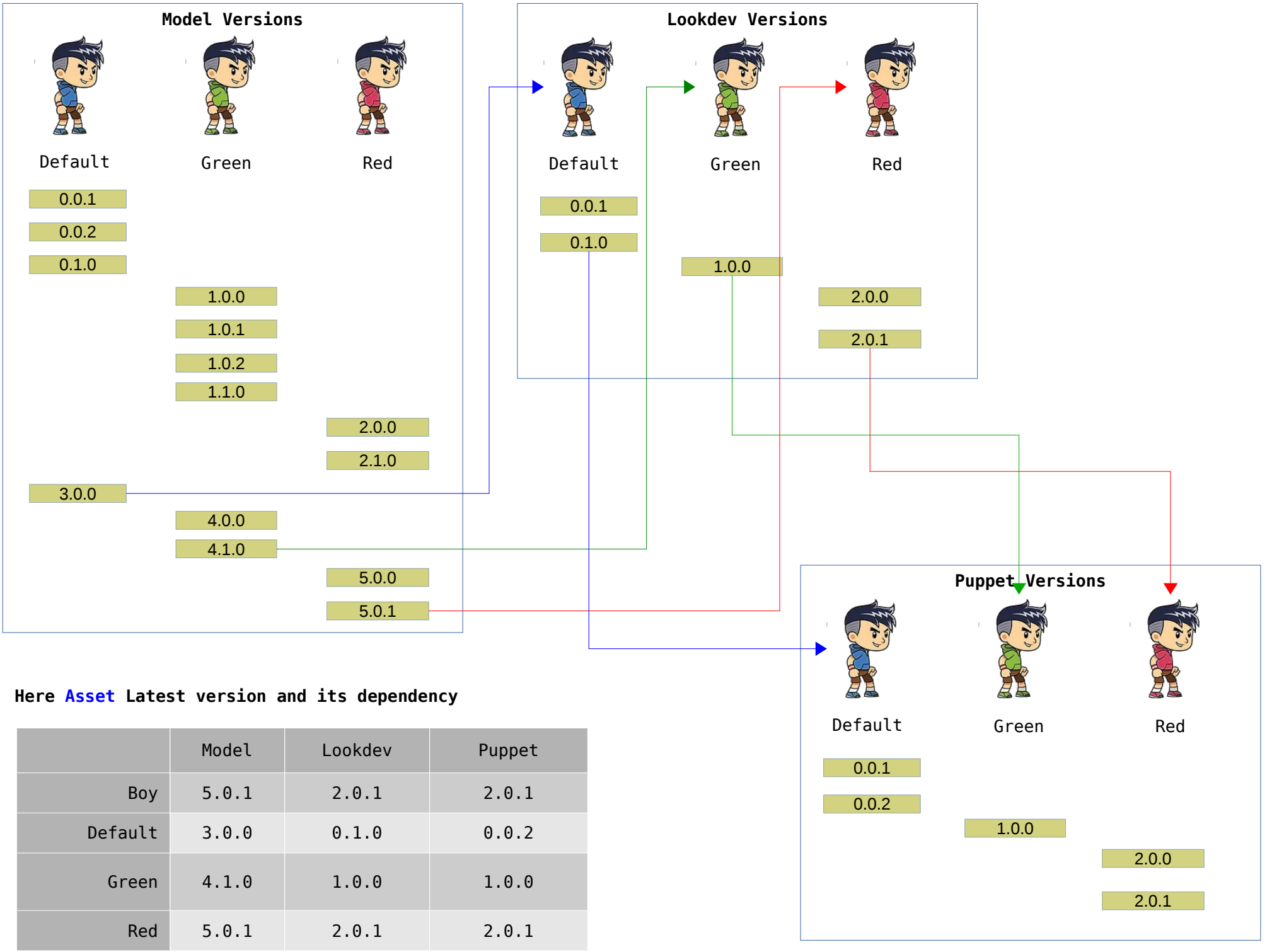
Here variants and versions we can include with semantic versions, Instead of v1, v2, v3, this is very efficient and visually understand what kind of changes happened, is it major or minor or patch.

The pipeline will provide APIs for creating, modify and query the assets or shots versions, latest versions, and their dependency versions.

- API/StudioAsset
- API/studioShot

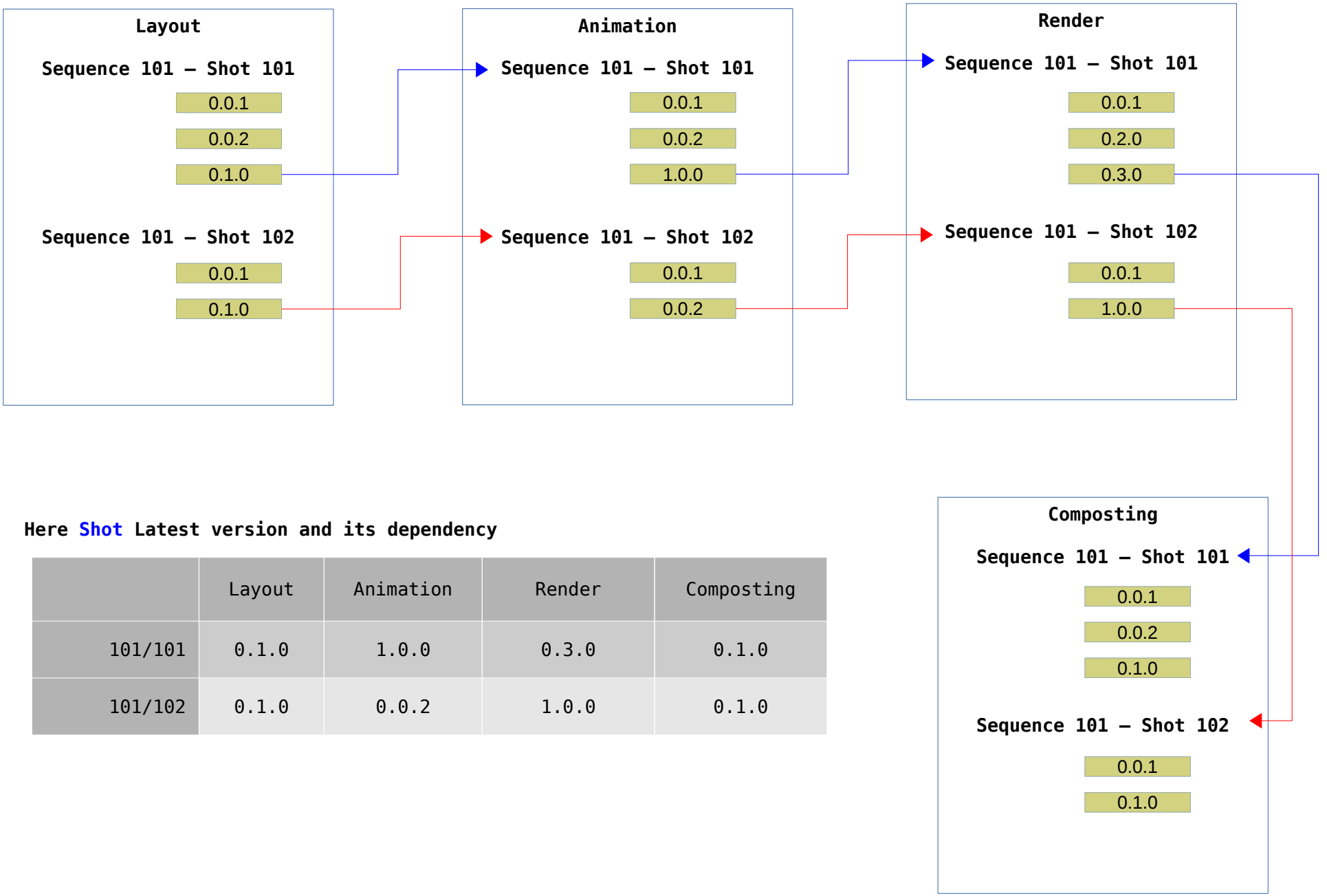
And any kind of versions and dependency conflict happened our tool will find out. Most probably conflict never happened because publish and build process done by Pipeline tools. And we have a version management tool as well.

Look at the below flowchart to get more ideas.



Here Asset Latest version and its dependency

	Model	Lookdev	Puppet
Boy	5.0.1	2.0.1	2.0.1
Default	3.0.0	0.1.0	0.0.2
Green	4.1.0	1.0.0	1.0.0
Red	5.0.1	2.0.1	2.0.1



Here Shot Latest version and its dependency

	Layout	Animation	Render	Composting
101/101	0.1.0	1.0.0	0.3.0	0.1.0
101/102	0.1.0	0.0.2	1.0.0	0.1.0

9. Step Published Data Types

Here, the output data might be change as per our requirement, I add some basic output data for the respective steps.

Steps	Outputs
Model + UV	Model Scene, UV Data, Alembic Cache, Scene Description
LookDev	Lookdev Scene, 4k and 0.5k Source Images, Render Presets, Scene Description
Puppet	Puppet Scene, Scene Description
Groom	Groom Scene, Scene Description
Layout	Layout Scene, Animation Data, Asset Scene Description, Movie, Scene Description
Animation	Animation Scene, Animation Data, Asset Scene Description, Animation Cache, Movie, Scene Description
Render	Render Scene, Asset Scene Description, Render outputs, Movie, Scene Description
Groom FX	Groom Scene, Groom Cache, Movie, Scene Description
FX	EFX Scene, EFX Cache, Movie, Scene Description
Composting	Composting Scene, Output movie, Scene Description

10. Core Pipeline Production Tools

- **Submit Tool**
  Submit the step task for approval, this is almost similar to a publishing concept, the submitted data is only for review proposes. We can call unregistered published data.

- **Publish Tool**
  Publish the step task to downstream steps. Registered show (project) data.

- **Build Tool**
  Prepare current step scene for the artist or user based on the task. The inputs are coming from respective upstream steps published data.

- **Version Management Tool**
  After building the scene, this tool can manage to set the appropriate version or the user has the option to chose what version want. Most probably its happened while any updates happen on the upstream steps publish.

We need to develop these four tools first (High Priority). After that, we can go to the development of tools and Automation.

11. Pipeline Resources

- **Git**
- **PyCharm or Eclipse**
- **Jira/Bugzilla/Redmine or any other issue tracking capabilities applications**
- **Wiki for documentation**
- **Python 2 or 3**
- **PySide**
- **MySQL or Shotgun data base.**

12. Pipeline Frameworks and Packages (Repository)

To address inconsistencies and duplicated development efforts, in our case to develop a production pipeline. Each framework has its classes contained by packages. All frameworks are each repository and frameworks are no dependent on any other repositories. Frameworks are standalone packages. And all repositories are dependent on the framework but and no dependency on other repositories.

which you are going to see the examples

1. **Studio API Framework** - core API for create, edit and query, assets, shots

   Asset API, Shot API, Render API, Publish API, DataBase API
2. **Studio Pipe Framework** - create and configure the show, DCCs, and studio applications
3. **Studio Core Framework** - Interacting with the file system, such as files, directories, dealing with time, date,

   platform data type, and system-related functionality and implementing.
4. **Model Repository**
5. **Lookdev Repository**
6. **Puppet Repository**
7. **Animation Repository**
8. **Render Repository**
9. **Composting Repository**

13. Mailing and Security

IT Department need to take care of both. But mailing - pipeline tools are connected, so each valid process such as publish, etc respective team get the notification emails.

14. Example of Automation that connect with two dependency steps

How to run more efficient, profitable, and time shows(projects)?

So our core Pipeline tools are designed based on these two facts. From the user perspective data transfer from one step to another is a simple and easy manner, with no confusion, just follow pipeline guide lines. On top of core tools, we can provide automation tools and libraries, such as rig builder, Blendshape tool, model, shader, pose, animation, and lookdev/render libraries. Any tools are required from the user, please share with us, we can implement them on the production.

For Example, I am going to demonstrate how to help automation in render?

In our one-show light modes and style we setup into 5 types, ie “day light”, “mid night”, “moon night”, “evening” and “morning”. And we have all shot dependency information. So while create render the scene, our build tool will bring all dependency data include light preset as well. At the same time, we can provide the Light Library tool as well, so after building the scene, the user can update the lights. So the advantage is lighter no need to start from the initial stage, one basic light is ready for then. And they can publish there on light preset as well.

This is one of the automation, like this way we can try to automate most of the steps, if any reuse is required.

Look at the below flowchart to get more ideas.

Sequence	Shot	Characters	Props	Sets	Light	Shadow Angle
101	101	Boy, Girl	Bat, Ball	Payground	Day Light	30*
	102	Boy, Girl	Bat, Ball	Payground	Day Light	30*
	103	Girl, Mom	Ball, Car	Girl House Ext	Evening	50*
102	101	Papa, Boy	Bat, Car	Boy House Ext	Night	45*
	102	Papa, Boy	Table, Computer	Boy House Int	Night	45*